Bluetooth Beacons for IOT Applications?

Mike Vartanian

7 August 2018

https://github.com/mvartani76/iot-detroit-august2018

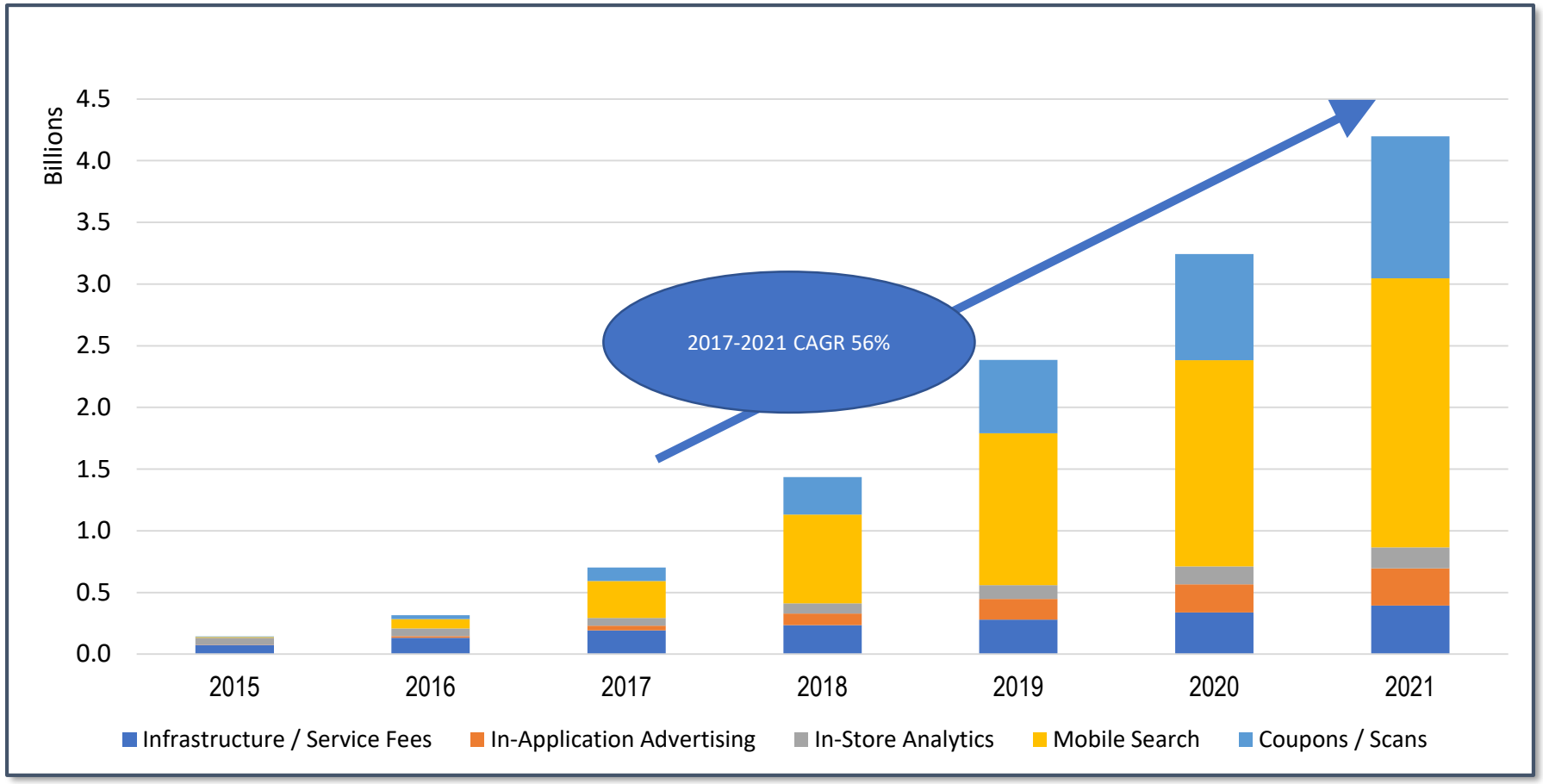
Agenda

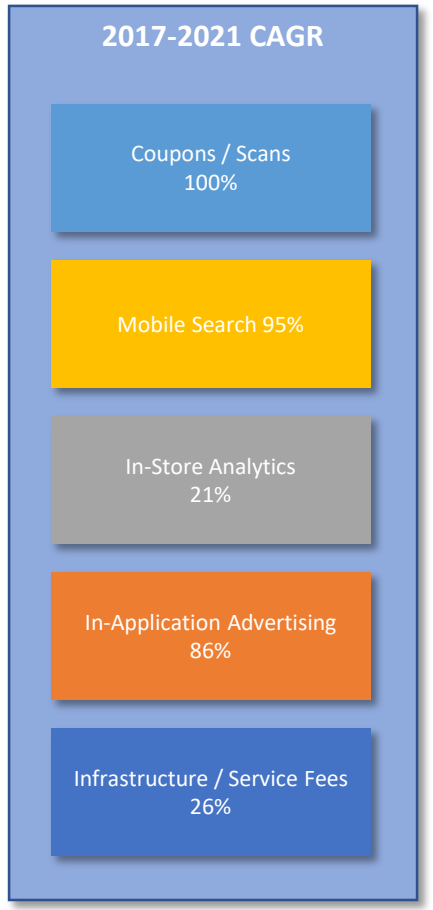
- Introduction
- Bluetooth Beacon Overview
  - What are beacons?
  - Competitive Landscape
  - Protocols
- Bluetooth Beacon Applications
  - Marketing/Advertising
  - IOT?
- Example Platforms
- Hands On Projects
  - Beacon Emulator (Raspberry Pi, iOS)
  - Beacon Scanner (iOS, Android)

What are Bluetooth Beacons?

- Bluetooth Beacons are low-cost, low-power transmitters equipped with Bluetooth Low Energy or BLE (also called Bluetooth 4.+ or Bluetooth Smart)
- A beacon transmits signals which allow another device to determine its proximity to the broadcaster
- The beacon does not transmit content

Is there a market for Bluetooth Beacons?





ABIresearch 2017

Are there many companies in this space?

Are there any Beacon standards?

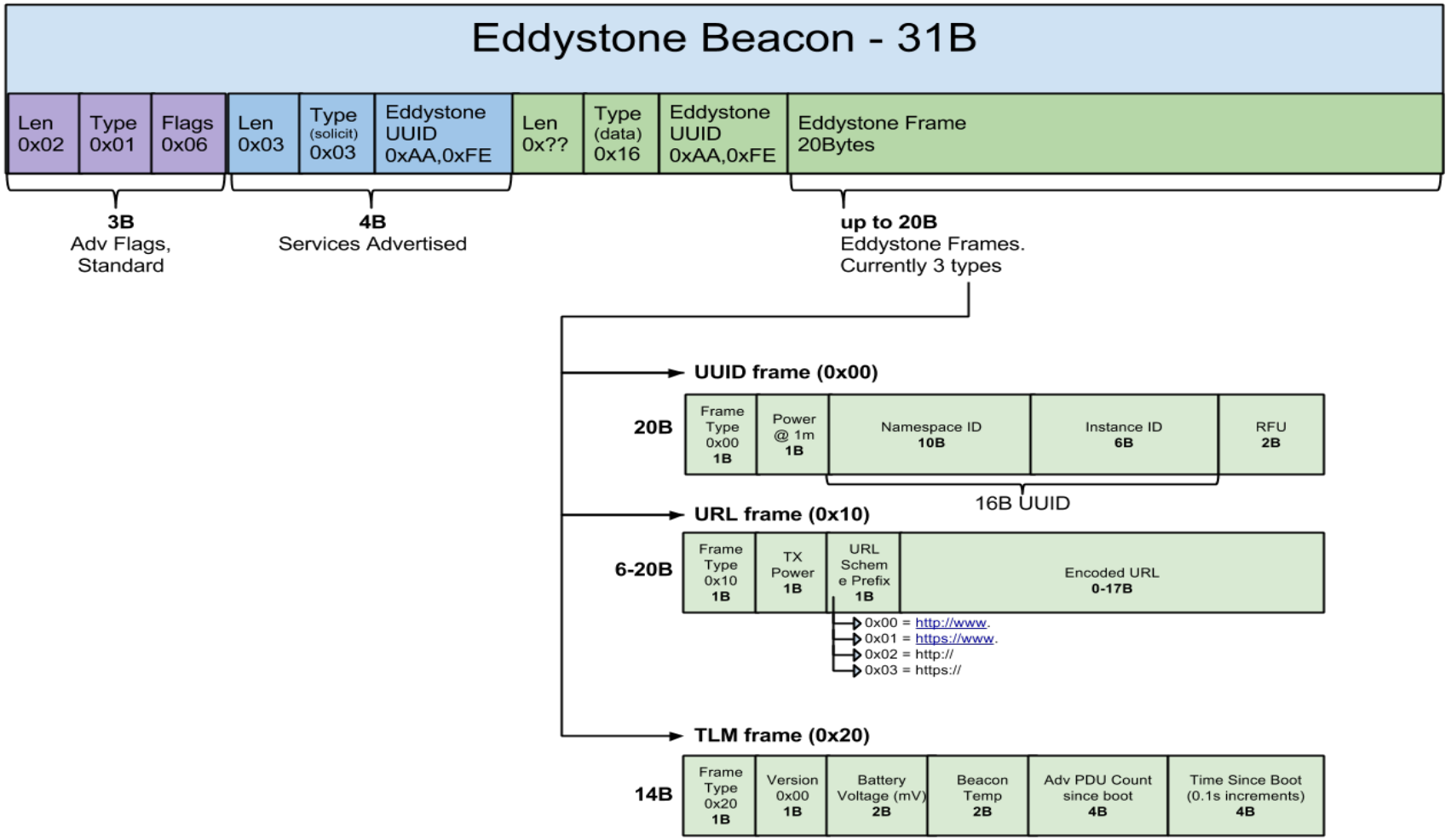
- iBeacon introduced in 2013 by Apple
  - Proprietary
- Eddystone introduced in 2015 by Google
  - Open source
- ALTBeacon introduced in 2014 by Radius Networks
  - Open source

What is Eddystone/Physical Web?

Physical Web

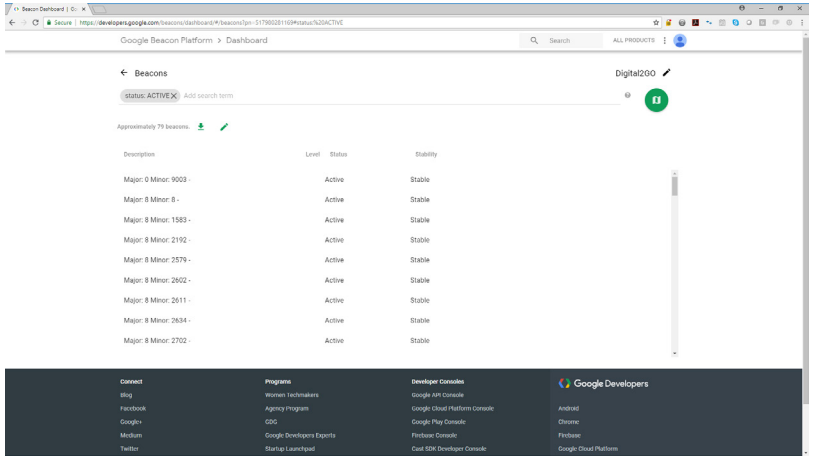
- Three payload types (Eddystone URL, Eddystone UID (unique ID), and Eddystone TLM (telemetry information)
- No app requirement in Android/iOS
  - Widgets
- Eddystone Beacons and Chrome Browser are the backbone of the Physical Web
- Google stopped support of Physical Web

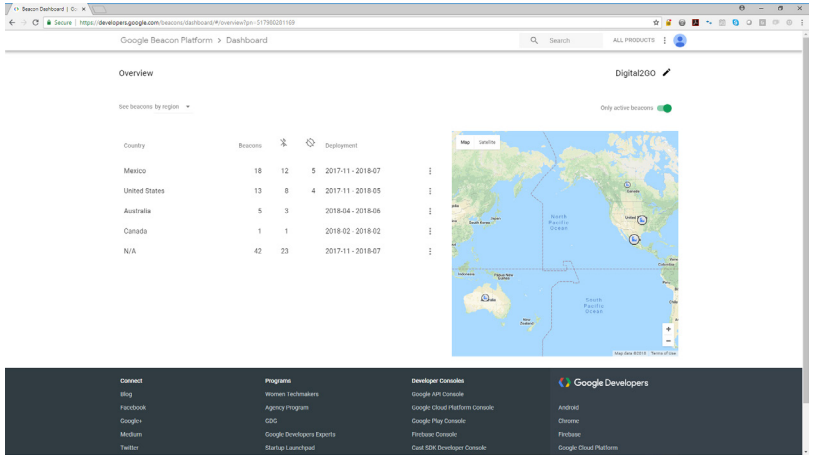
Eddystone Protocol



What are Nearby Notifications?

- Nearby Notifications supported on Android 4.4 (KitKat) and newer
- Not currently supported in iOS
- Nearby Notifications helps users to discover what's around them
  - Surfacing location-specific notifications for apps and websites
  - No prior app install required
- Works with Eddystone and iBeacon
- Two basic experience types
  - Link to an **HTTPS** URL
  - Trigger an app intent



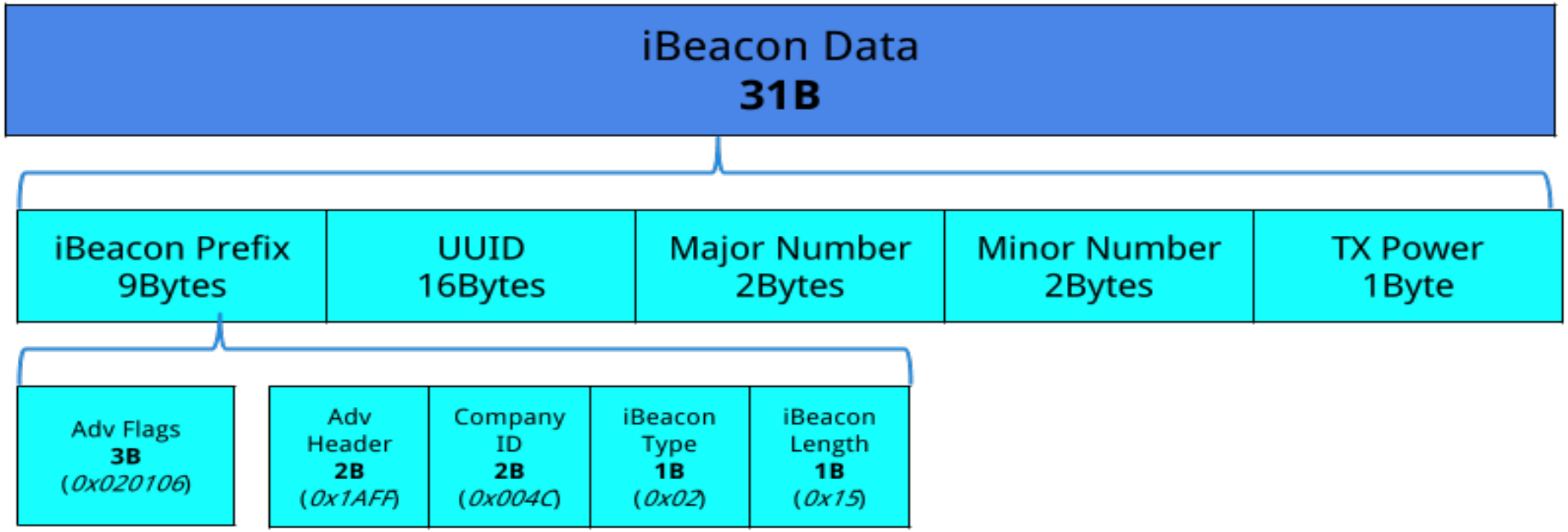


https://developers.google.com/beacons/dashboard

What is iBeacon?

- iBeacon is the name of an Apple technology standard introduced in 2013 with iOS 7
- Both iOS and Android can recognize iBeacons
- Utilizes Bluetooth Low Energy (BLE) advertising introduced in BT4.0
- Systems usually consist of broadcaster/transmitter and observer/receiver (mobile app)

iBeacon Protocol



1.A UUID that identifies the beacon.
2.A Major number identifying a subset of beacons within a large group.
3.A Minor number identifying a specific beacon.
4.A TX power level in 2's compliment, indicating the signal strength one meter from the device. This number must be calibrated for each device by the user or manufacturer.

iBeacon Protocol

- Must transmit UUID, Major, and Minor to be iBeacon compliant

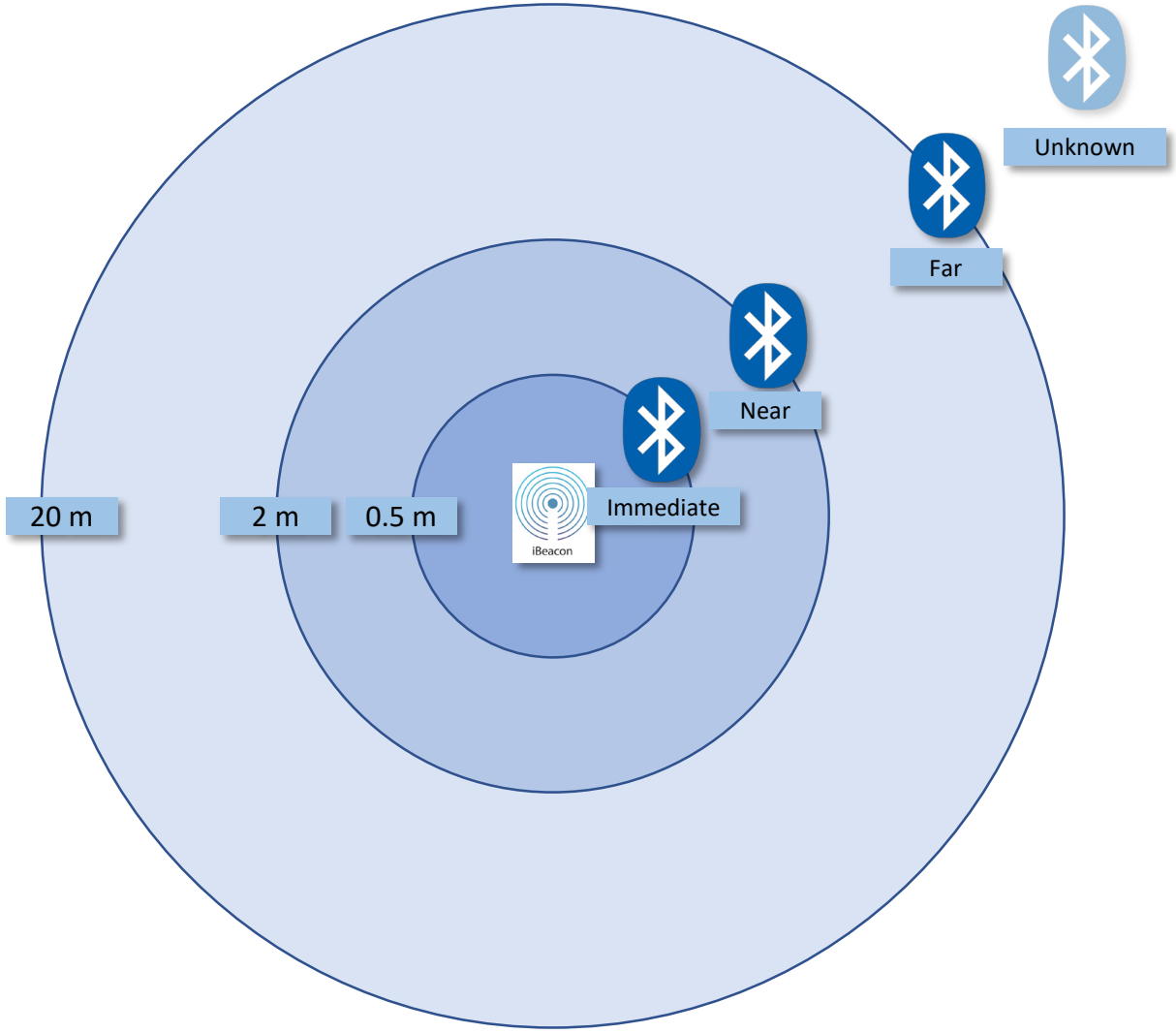
Field	Size	Description
UUID	16 bytes	Universal Unique Identifier. Application developers should define a UUID specific to their app and deployment use case.
Major	2 bytes	Further specifies a specific iBeacon and use case. For example, this could define a sub-region within a larger region defined by the UUID.
Minor	2 bytes	Allows further subdivision of region or use case, specified by the application developer.

- Retail Example

Store Location		Detroit	Ann Arbor	Troy
UUID		8D847D20-0116-435F-9A21-2FA79A706D9E		
Major		1	2	3
Minor	Clothing	10	10	10
	Grocery	20	20	20
	Auto	30	30	30

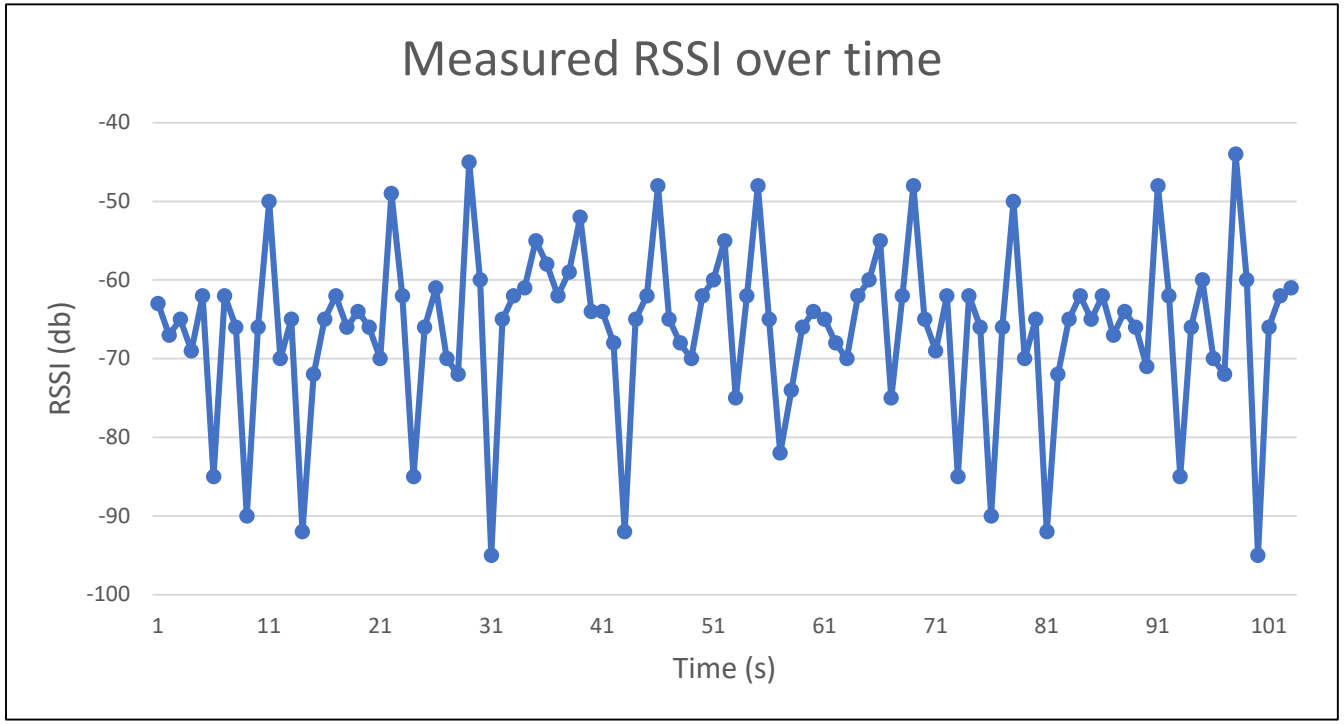
What is Proximity?

- iBeacon is the name of an Apple technology standard introduced in 2013 with iOS 7
- Both iOS and Android can recognize iBeacons
- Utilizes Bluetooth Low Energy (BLE) advertising introduced in BT4.0
- Systems usually consist of broadcaster/transmitter and observer/receiver (mobile app)
- Based off of Received Signal Strength Indicator (RSSI)



Is RSSI a reliable indicator for Proximity?

- Estimate distance from RSSI
- RSSI has a lot of variance
  - Attenuation from people and objects (especially metal)
- Kalman Filter could remove some noise
  - Will add latency though



$$RSSI(dBm) = -10 * (Propagation\ Constant\ n) * \log_{10}(distance) + (dBm\ @1m)$$

$$Propagation\ Constant\ (Path\ Loss) in\ free\ space = 2$$

$$distance(m) = 10^{\frac{(dBm\ @\ 1m) - RSSI}{10 * n}}$$

How do I interact with Beacons?

Monitoring

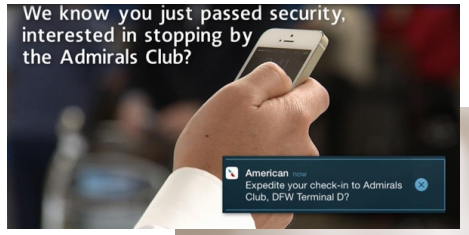
- Actions triggered on entering/exiting region's range
- Works while app is running, suspended, or killed
- Not as responsive as ranging – uses low power scanning
- iOS limits number of simultaneous monitored regions to 20

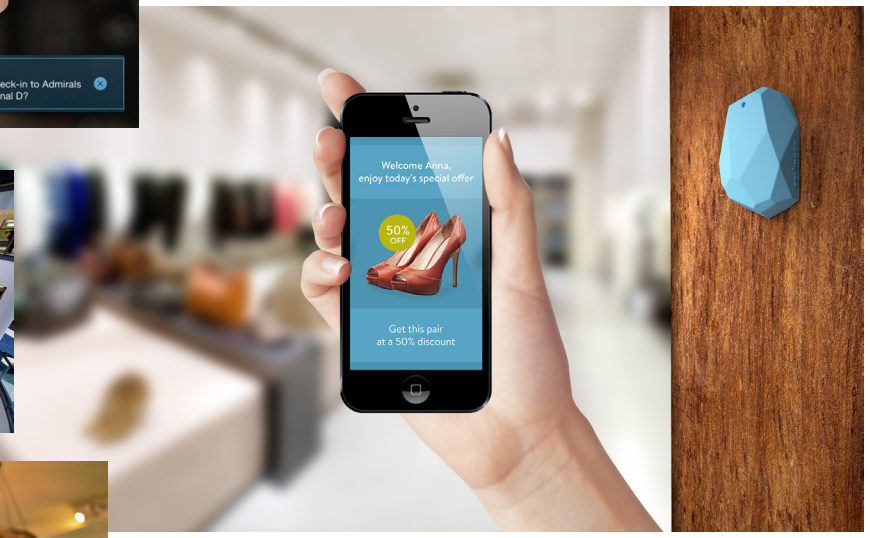
Ranging

- Actions triggered based on proximity to a beacon
- Returns RSSI → proximity, distance
- Only works while app is running…. Well not exactly true…

What are typical beacon use cases?

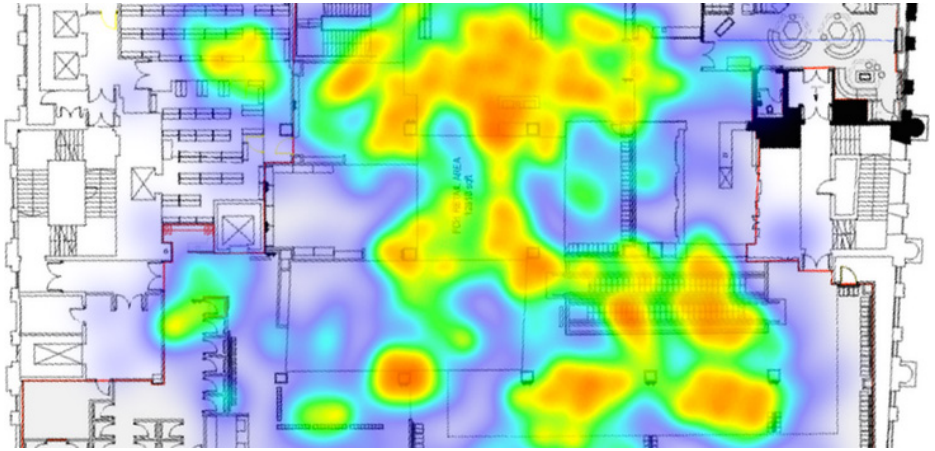
- Advertising/Marketing
  - Coupons
- Customer Check-Ins
- Mobile Payment
- Indoor Wayfinding
- Marketing Intelligence
- Enhanced Customer Experience
  - Spot Info
  - Digital Signage





Is there a place for beacons in IOT?

- Digital to Physical Linkage
- Access Control
  - Rooms
  - Equipment
- Asset Tracking
- Workforce Monitoring
- Indoor Wayfinding
- Content Delivery



So how do beacons work in these scenarios?

1

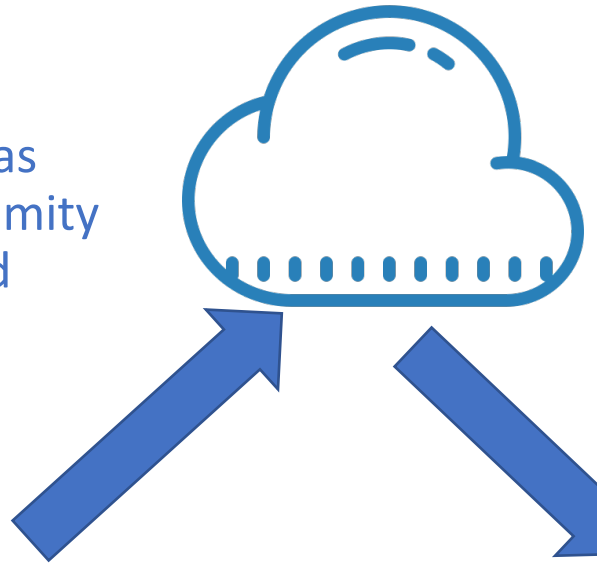
Bluetooth Beacon continuously advertising identity

2

Smartphone with appropriate Bluetooth enabled mobile app comes into proximity to beacon

3

Smartphone acting as gateway sends proximity information to cloud



4

Cloud platform logs data and determines what actions to take

5

Smartphone app receives appropriate content/trigger

Example Platforms

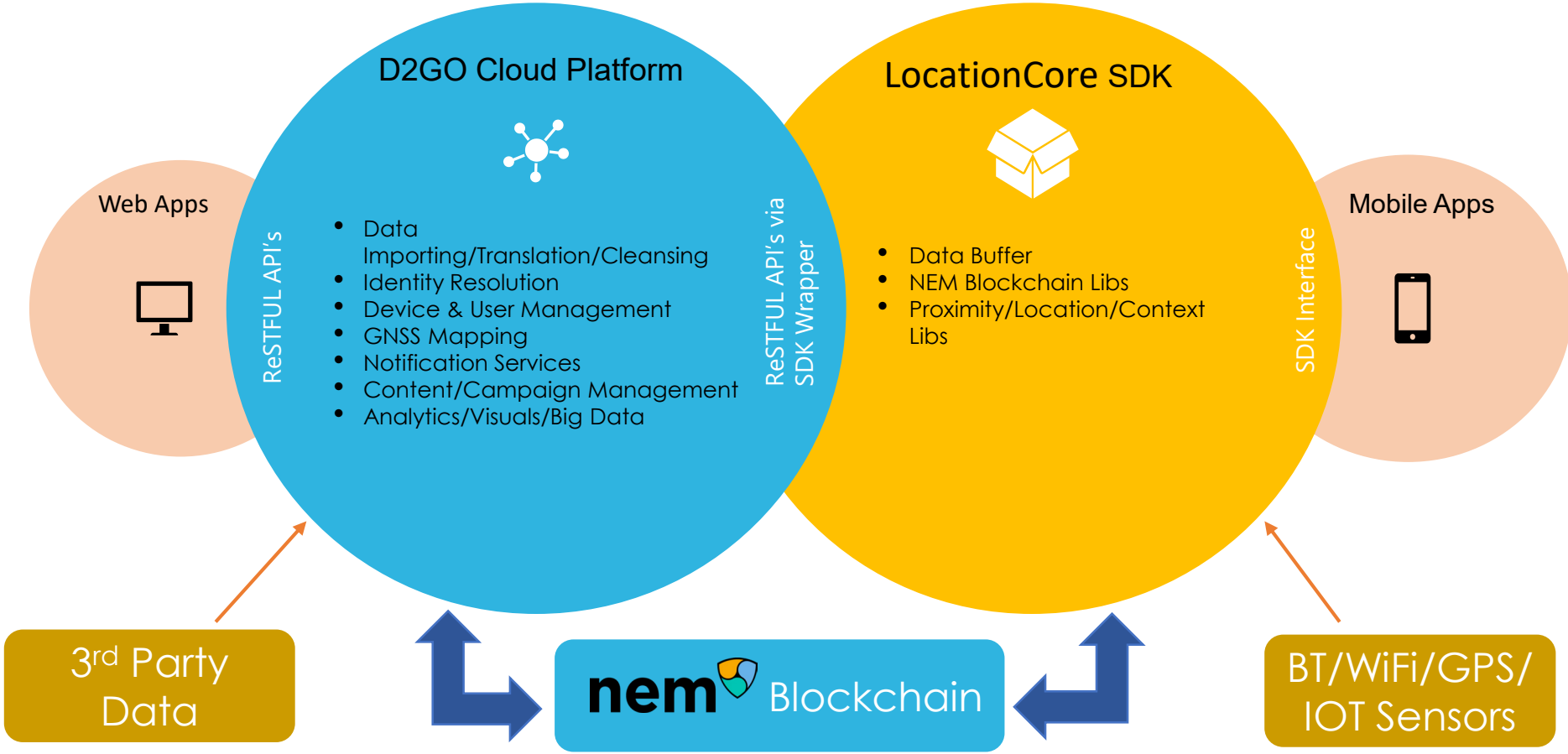
Digital2Go Media Networks

Sahuaro Labs

Who are Digital2Go Media Networks?

- Digital2Go Media Networks (http://www.digital2go.com/) is a consumer engagement and marketing intelligence company.
  - SDKs
  - 25B+ mobile audience data records
  - Blockchain

D2GO Platform/SDK Powers Full Stack Solution



Who are Sahuaro Labs?

sahuarolabs

- Sahuaro Labs (http://www.sahuarolabs.com/) is a mobile and web development company specializing in IOT:
  - Location services (Proximity)
  - App Development with beacon interaction
  - Other Sensors

Example Project – Caffenio Drive

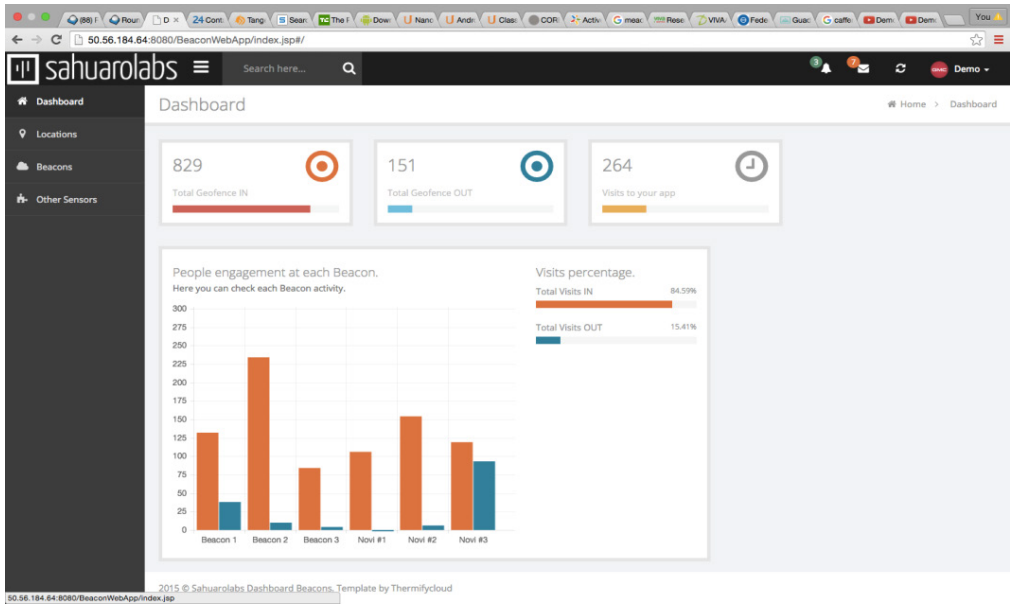
- Caffenio Drive is a regional coffee chain in Northern Mexico, which offers coffee, food and other cold beverages on their drive thru stores with fast and reliable service
- Proximity Services to improve business processes and customer experience

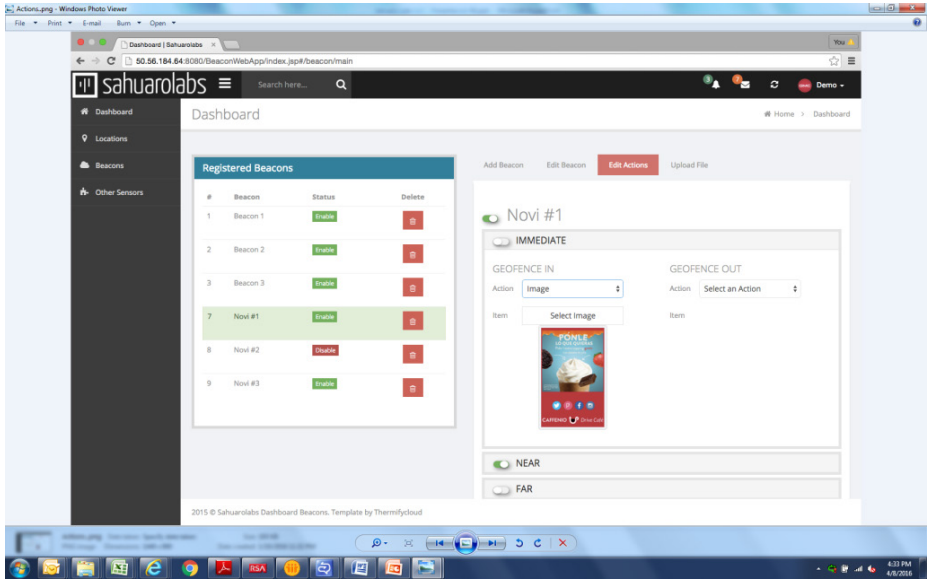
YouTube Video
https://youtu.be/o-mpJT_g904

Results in a cashless and seamless transaction

IoT Sensor Platform

- We are developing an IOT platform focusing on sensors:
- Proximity (ibeacons)
- Process Control (examples) – [Beta]
  - Temperature
  - Gases
  - HS&E
  - Adding more…





Can be utilized to monitor processes, reduce costs and improve efficiency

Smart Building, Factory or Process

Hands On Projects

Bluetooth Beacon Emulator

Bluetooth Beacon Scanner

Hands On Project –Beacon Emulator - RPi

- Use BlueZ for Linux Bluetooth Beacon
  - BlueZ is the official Linux Bluetooth protocol stack
  - BlueZ **5.43** comes preinstalled on Raspbian-Stretch
  - Latest release of BlueZ is **5.50** as of **3 June 2018**
- Install/Update BlueZ
- Verify Functionality
- Example Code Overview

Install BlueZ on Raspberry Pi

- Install Dependencies
  - Update Packages

    ```
    sudo apt-get update
    ```
  - Install the Dependencies

    ```
    sudo apt-get install libdbus-1-dev libglib2.0-dev
    libudev-dev libical-dev libreadline-dev -y
    ```

Install BlueZ on Raspberry Pi

- Install Latest BlueZ
  - Download the latest version of BlueZ source code

    ```
    wget www.kernel.org/pub/linux/bluetooth/bluez-5.50.tar.xz
    ```
  - Extract the downloaded file and navigate to BlueZ Directory

    ```
    tar xvf bluez-5.50.tar.xz
    cd bluez-5.50
    ```
  - Configure

    ```
    ./configure --prefix=/usr --mandir=/usr/share/man --sysconfdir=/etc --localstatedir=/var     --enable-experimental
    ```
  - Compile and Install

    ```
    make –j4
    sudo make install
    ```
  - Reboot

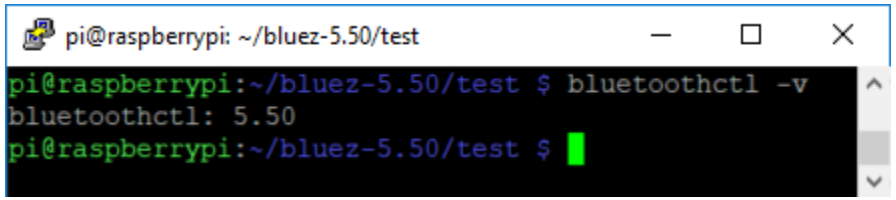
    ```
    sudo reboot
    ```

https://raspberrypi.stackexchange.com/questions/66540/installing-bluez-5-44-onto-raspbian#74712
https://scribles.net/updating-bluez-on-raspberry-pi-5-43-to-5-48/
https://scribles.net/creating-ibeacon-using-bluez-example-code-on-raspberry-pi/

Verify Correct BlueZ Installation

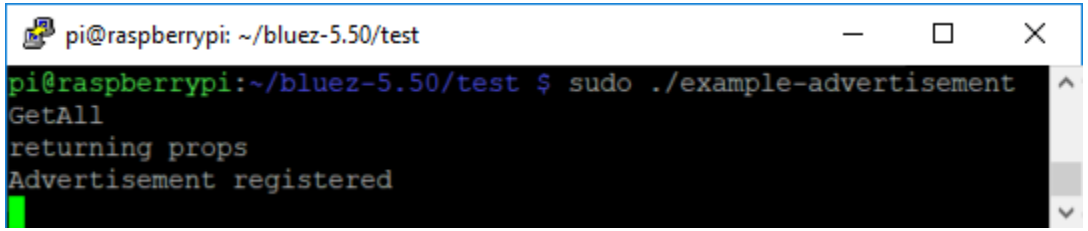
- Install Latest BlueZ
  - Verify Correct BlueZ Version

```
bluetoothctl -v
```



  - Verify that sample advertisement code works

```
sudo ./bluez-5.50/test/example-advertisement
```



Bluetooth iBeacon Emulator Code Overview

- Python Script continuously sends Bluetooth advertising packets
  - **hcitool** is used to configure Bluetooth connections and send some special command to Bluetooth devices

```
Usage: sudo ibeacon.py [-u|--uuid=UUID or `random' (default=Digital2Go Media Networks UUID)]
                       [-M|--major=major (0-65535, default=0)]
                       [-m|--minor=minor (0-65535, default=0)]
                       [-a]--minadv=minimum advertising interval (100-3000ms, default=350ms)]
                       [-A]--maxadv=maximum advertising interval (100-3000ms, default=400ms)]
                       [-p|--power=power (0-255, default=200)]
                       [-d|--device=BLE device to use (default=hci0)]
                       [-z|--down]
                       [-v|--verbose]
                       [-n|--simulate (implies -v)]
                       [-h|--help]
```

```
0x08: BLE Group
0x0008: Set LE Advertising Data
0x0006: Set LE Advertising Parameters
0x000a: Set LE Advertising Enable
```

iBeacon Protocol

```
process_command("hcitool -i hci0 cmd 0x08 0x0008 1E 02 01 1A 1A FF 4C 00 02 15 %s %s %s %s 00
>/dev/null" % (split_uuid, split_major_hex, split_minor_hex, power_hex)

process_command("hcitool -i hci0 cmd 0x08 0x0006 %s %s 03 00 00 00 00 00 00 00 00 07 00")

process_command("hcitool -i hci0 cmd 0x08 0x000a 01")
```

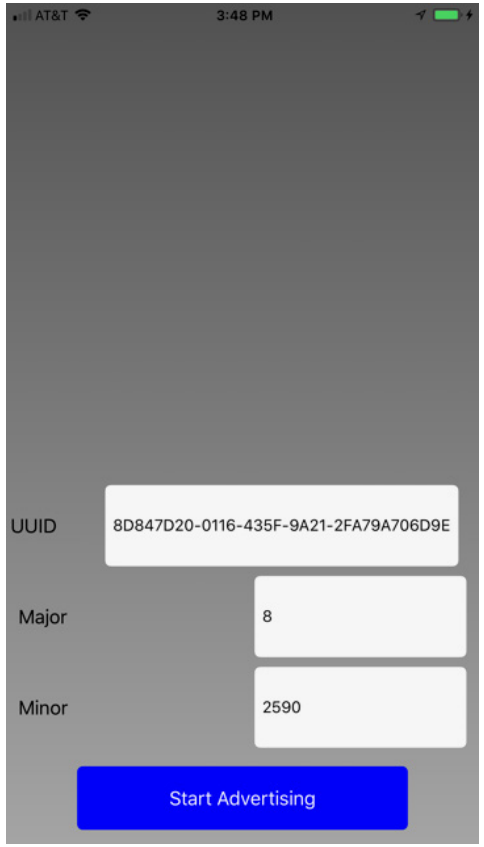
Min Advertising Interval
00 A0 * 0.625 = 100ms

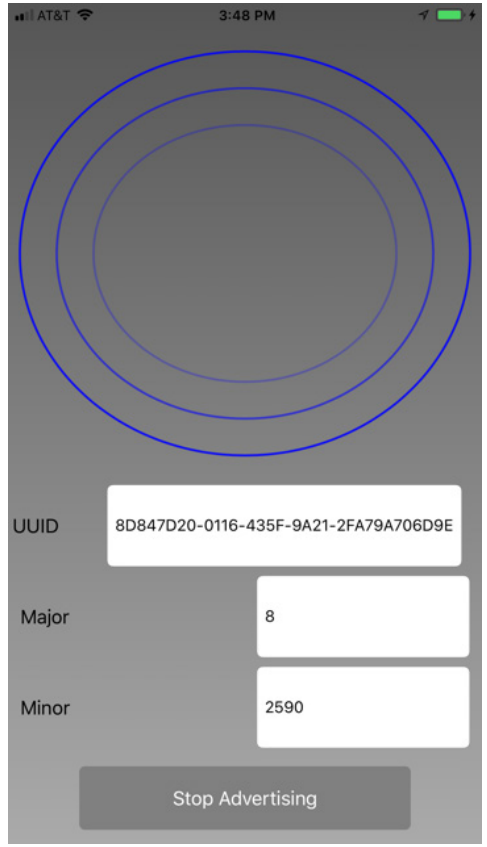
Max Advertising Interval
00B0 * 0.625 = 110ms

https://stackoverflow.com/questions/21124993/is-there-a-way-to-increase-ble-advertisement-frequency-in-bluez
https://stackoverflow.com/questions/35872097/advertise-bluetooth-le-service-using-hcitool
https://esf.eurotech.com/docs/how-to-user-bluetooth-le-beacons
https://github.com/dburr/linux-ibeacon
http://dev.ti.com/tirex/content/simplelink_cc2640r2_sdk_1_35_00_33/docs/ble5stack/ble_user_guide/html/doxygen/group___h_c_i.html#gaf75bbdb11a019fa4183379f7d9335942

Hands On Project –Beacon Emulator - iOS

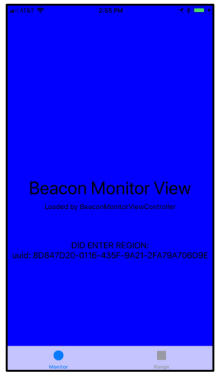
- Code pseudo emulates the iBeacon protocol
  - Configure UUID, Major, Minor

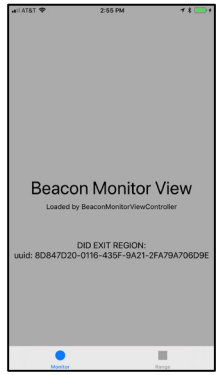




iOS Bluetooth Beacon Scanner Code Overview

- Multi-Tab Application
  - Monitoring Tab
    - Monitors for Region changes
    - Changes screen to blue if ENTER, light gray if EXIT
    - Not as responsive as Ranging (~5s for ENTER, ~30s for EXIT)
    - Lots of complaints actually → Estimote Monitoring
  - Ranging Tab
    - Provides further information about Beacons
      - Region, Proximity, RSSI
    - Changes screen color based on Proximity





iOS Bluetooth Beacon Scanner Code Overview

BeaconMonitorViewController.Swift

```
override func viewDidAppear(_ animated: Bool) {
        if let uuid = NSUUID(uuidString: IBEACON_PROXIMITY_UUID) {
            let beaconRegion = CLBeaconRegion(proximityUUID: uuid
              as UUID, identifier: "iBeacon")
              startMonitoring(beaconRegion: beaconRegion)
              monitorLabel.text = ""
        }
    }
```

```
func locationManager(_ manager: CLLocationManager, didEnterRegion
region: CLRegion) {
        if let beaconRegion = region as? CLBeaconRegion {
             monitorLabel.text = "DID ENTER REGION: \nuuid:
             \(beaconRegion.proximityUUID.uuidString)"
        }
    }
```

```
func locationManager(_ manager: CLLocationManager, didExitRegion
region: CLRegion) {
        if let beaconRegion = region as? CLBeaconRegion {
             monitorLabel.text = "DID EXIT REGION: \nuuid:
             \(beaconRegion.proximityUUID.uuidString)"
        }
    }
```

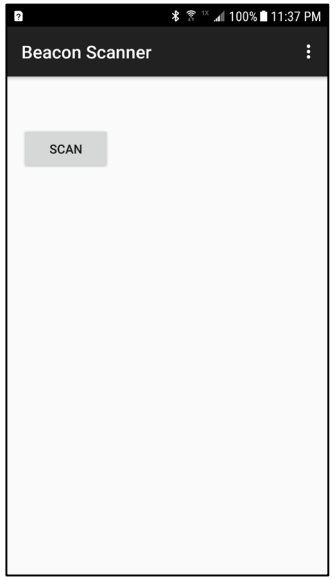
BeaconRangeViewController.Swift

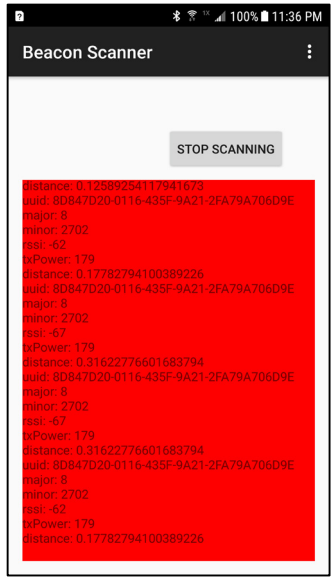
```
override func viewDidAppear(_ animated: Bool) {
       if let uuid = NSUUID(uuidString: IBEACON_PROXIMITY_UUID) {
           let beaconRegion = CLBeaconRegion(proximityUUID: uuid as UUID,
             identifier: "iBeacon")
             startRanging(beaconRegion: beaconRegion)
             rangeLabel.text = ""
             rangeView.backgroundColor = UIColor.white
       }
   }
```

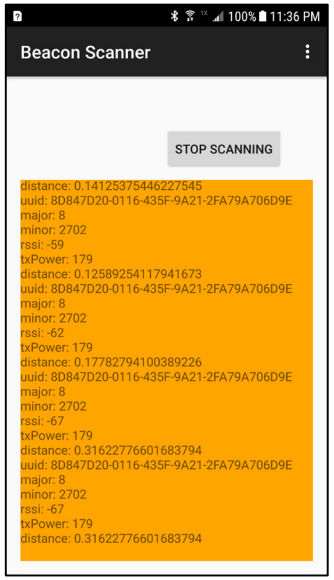
```
func locationManager(_ manager: CLLocationManager, didRangeBeacons beacons: [CLBeacon], in
region: CLBeaconRegion) {
       for beacon in beacons {
           var beaconProximity: String;
           switch (beacon.proximity) {
           case .unknown:
               beaconProximity = "Unknown"
               rangeView.backgroundColor = UIColor.white
           case .immediate:
               beaconProximity = "Immediate"
               rangeView.backgroundColor = UIColor.red
           case .near:
               beaconProximity = "Near"
               rangeView.backgroundColor = UIColor.orange
           case .far:
               beaconProximity = "Far"
               rangeView.backgroundColor = UIColor.yellow
           }
              rangeLabel.text = "BEACON RANGED:\nuuid:
              \(beacon.proximityUUID.uuidString)\nmajor: \(beacon.major)\nminor:
              \(beacon.minor)\nproximity: \(beaconProximity)"
       }
   }
```

Android Bluetooth Beacon Scanner Code Overview

- Not as easy as iOS
  - No built in methods
    - monitoring or ranging
    - Proximity
- Single Tab Application
  - Scans BLE Advertisements
  - Extracts iBeacon information
  - Changes TextView color based on Proximity
  - Computes Proximity based on estimated distance
  - Distance computed from RSSI







Android Bluetooth Beacon Scanner Code Overview

MainActivity.java

```
final byte[] scanRecord = result.getScanRecord().getBytes();

int startByte = 2;
boolean patternFound = false;
while (startByte <= 5) {
if (((int) scanRecord[startByte + 2] & 0xff) == 0x02 && //Identifies an iBeacon
     ((int) scanRecord[startByte + 3] & 0xff) == 0x15) { //Identifies correct data length
     patternFound = true;
     break;
     }
startByte++;
}

if (patternFound) {
     //Convert to hex String
     byte[] uuidBytes = new byte[16];
     System.arraycopy(scanRecord, startByte + 4, uuidBytes, 0, 16);
     String hexString = bytesToHex(uuidBytes);

     //UUID detection
     String uuid = hexString.substring(0, 8) + "-" +
     hexString.substring(8, 12) + "-" +
     hexString.substring(12, 16) + "-" +
     hexString.substring(16, 20) + "-" +
     hexString.substring(20, 32);

     final int major = (scanRecord[startByte + 20] & 0xff) * 0x100 + (scanRecord[startByte + 21] & 0xff);
     final int minor = (scanRecord[startByte + 22] & 0xff) * 0x100 + (scanRecord[startByte + 23] & 0xff);
     final int tx_power = (scanRecord[startByte + 24] & 0xff);
```

```
 double getDistance(int rssi, int txPower) {

        return Math.pow(10d, ((double) txPower - rssi) / (10 * 2));
    }
```

```
  String getProximity(double distance) {
        if (distance < PROXIMITY_IMMEDIATE_THRESH)
            return "immediate";
        else if (distance < PROXIMITY_NEAR_THRESH)
            return "near";
        else if (distance < PROXIMITY_FAR_THRESH)
            return "far";
        else
            return "unknown";
    }
```

```
if (uuid.equals(d2go_uuid)) {
     // it appears that txpower numbers are 8 bit 2s complement
     final double distance = getDistance(result.getRssi(), tx_power-256);
     final String proximity = getProximity(distance);
     switch(proximity) {
          case "immediate":
                peripheralTextView.setBackgroundColor(Color.RED);
                break;
          case "near":
                peripheralTextView.setBackgroundColor(Color.rgb(255,165,0));
                break;
          case "far":
                peripheralTextView.setBackgroundColor(Color.YELLOW);
                break;
          case "unknown":
                peripheralTextView.setBackgroundColor(Color.WHITE);
                break;
          default:
                peripheralTextView.setBackgroundColor(Color.WHITE);
                return;
     }
```